O'REILLY®

Vue.js

kurz & gut

O'REILLYS TASCHENBIBLIOTHEK

Lars Peterke

Vue.js

kurz & gut

Lars Peterke

O'REILLY®

Lars Peterke

Lektorat: Ariane Hesse
Fachgutachter: Joe Ray Gregory, Alexander Schwartz, Martin Wohlrab
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Satz: III-satz, www.drei-satz.de
Herstellung: Stefanie Weidner
Umschlaggestaltung: Michael Oréal, www.oreal.de
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

ISBN:
Print 978-3-96009-092-2
PDF 978-3-96010-246-5
ePub 978-3-96010-247-2
mobi 978-3-96010-248-9

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

1. Auflage

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Inhalt

Teil II: Webentwicklung mit Vue

Teil III: Vue-API

Teil IV: Tägliches Arbeiten mit Vue

Vorwort

Die Anforderungen an moderne Webseiten und Applikationen auf Basis von Webtechnologien sind in den letzten Jahren immens gestiegen. Frontend-Bibliotheken wie *jQuery* (2006) reichen heute nicht mehr aus, um den gestiegenen Anforderungen an Benutzeroberflächen und deren Bedienbarkeit (UI/UX) gerecht zu werden.

In den letzten Jahren taten sich viele JavaScript-Frameworks hervor, die diesen Anforderungen mit strukturierten Ansätzen zur komponentenbasierten Entwicklung und zum Templating gerecht werden wollten. Zu den bekanntesten Frameworks dieser Art zählen *React*, *Angular*, *Ember* und *Vue.js*, das oft einfach nur *Vue* genannt wird.

Unter allen JavaScript-Frameworks hat Vue in den letzten Jahren trotz seines noch jungen Alters die sicherlich beeindruckendste Karriere hingelegt. Und wenn Sie diese Zeilen lesen, haben Sie sich vermutlich dazu entschieden, Vue eine Chance zu geben, oder nutzen es bereits.

Was genau Ihre Motivation auch sein mag, mit dieser Taschenreferenz haben Sie den idealen Begleiter. »Vue.js kurz & gut« behandelt alle wichtigen Aspekte der Vue-Syntax. Damit ist dieses Werk zu gleichen Teilen Einstiegshilfe und Nachschlagewerk für die tägliche Arbeit mit Vue.

Das Buch behandelt Vue 2 in seiner aktuellen Version (2.5).

Der Aufbau dieses Buchs

Diese Kompakteinführung ist in vier Teile gegliedert. In Teil I, »Vue-Grundlagen«, werden alle grundlegenden Funktionen von Vue.js

erläutert. Teil II, »Webentwicklung mit Vue«, beschreibt den fortgeschrittenen Einsatz von Vue in Webprojekten mit komponentenbasierter Entwicklung. In Teil III, »Vue-API«, wird auf alle Aspekte der Vue-API eingegangen. Der abschließende vierte Teil, »Tägliches Arbeiten mit Vue«, liefert hilfreiche Informationen zu weiteren Bestandteilen von Vue und gibt einige Tipps zu typischen Aufgaben und Problemen im Arbeitsalltag.

Benötigte Grundlagen

Vue.js ist ein JavaScript-Framework zur Entwicklung moderner Benutzeroberflächen mit Webtechnologien. Sie sollten also gewisse Kenntnisse mitbringen, die in diesem Taschenbuch nicht vermittelt werden können.

Falls Sie bisher noch nie mit JavaScript gearbeitet haben, empfiehlt es sich, zunächst die Grundlagen dieser Sprache zu erlernen. Für die Gestaltung von Oberflächen im Web sind zudem Kenntnisse in der *Hypertext Markup Language* (HTML) sowie über die *Cascading Style Sheets* (CSS) notwendig. Für einige wenige Bereiche in diesem Buch ist es darüber hinaus von Vorteil, ein Grundwissen über modulare Build-Systeme wie npm, Babel und Webpack mitzubringen.

Zu allen genannten Themen sind eigene Werke aus der Reihe »kurz & gut« erhältlich. Für einen umfassenderen Einstieg in die einzelnen Themen eignen sich hingegen Bücher aus der Reihe »Von Kopf bis Fuß«, die ebenfalls im O'Reilly Verlag erschienen ist.

Um die Codebeispiele in diesem Buch zu testen, wird grundsätzlich nur ein einfacher Texteditor mit Syntax-Highlighting wie etwa *Sublime Text* sowie ein aktueller Webbrowser wie etwa *Google Chrome* benötigt.

Langfristig empfiehlt sich ein erweiterter Quelltexteditor wie etwa *Visual Studio Code* von Microsoft. Visual Studio Code ist für PC, Mac und Linux kostenfrei erhältlich. Der integrierte Marketplace hält bereits die nötigen Erweiterungen für die Arbeit mit Vue bereit. Als Alternativen zu Visual Studio Code sind *WebStorm* und *Atom* zu nennen.

Konventionen

Die Syntax für Skriptsprachen wie JavaScript wird von der *Ecma International* als *ECMAScript* spezifiziert. Um maximale Kompatibilität zu gewährleisten, folgen alle Codebeispiele der Syntax von ES5 (ECMAScript 2015). Sie sollten auch in älteren Browsern lauffähig sein.

Ferner werden die folgenden typografischen Konventionen verwendet:

Kursiv
: Kennzeichnet neue Begriffe, Eigennamen, Weblinks sowie Datei- und Pfadangaben.

`Nicht-Proportional-Schrift`
: Kennzeichnet Programmcode und programmspezifische Elemente wie Variablen oder Funktionsnamen in den Textabsätzen.

`Nicht-Proportional-Schrift fett`
: Kennzeichnet Hervorhebungen innerhalb von Codebeispielen oder markiert Befehle, die vom Benutzer einzugeben sind.

Tipps, Vorschläge und sonstige Anmerkungen werden mit diesem Symbol gekennzeichnet.

Danksagungen

Dieses Buch wäre sicherlich nicht ohne einige Personen aus der Laravel-Community entstanden. Dabei ist Taylor Otwell zu nennen, der mich schon früh auf Vue aufmerksam machte und dafür sorgte, dass ich bereits mit der Version 0.12 von Vue erste Experimente durchführte.

Jeffrey Way und Adam Wathan sind beide hervorragende Lehrer, die genau wie viele andere Autoren innerhalb der Vue-Community

mit allerhand Videotutorials, Artikeln und Tweets Dinge beigesteuert haben, die dieses Buch zu dem gemacht haben, was es nun ist.

Weiterer Dank gilt Ariane Hesse, allen weiteren Beteiligten von O'Reilly sowie den Testlesern Roman Arzaroli, Joe Ray Gregory, Alexander Schwartz und Martin Wohlrab.

Ein abschließender Dank geht an Matt Stauffer, der als ein erster Türöffner recht unwissentlich zur Entstehung dieses Buchs beigetragen hat.

Vue-Grundlagen

Los geht's! In diesem Teil machen wir uns mit den Grundlagen von Vue und der Syntax vertraut. Wir besprechen ein erstes Beispiel und lernen mit dem *Reactive Data Binding* eine Kernfunktion von Vue kennen.

Danach sehen wir uns an, wie Daten mit Vue dargestellt werden können. Wir lernen Templates und Kontrollstrukturen kennen und erweitern die von uns erzeugten Vue-Instanzen.

Zum Abschluss beschäftigen wir uns mit Events und Formularfeldern und erstellen erste interaktive Beispiele. Danach sind Sie bereits dazu in der Lage, etwa das HTML einzelner Seiten Ihrer bestehenden Webseite mit Vue aufzupeppen!

KAPITEL 1

Einführung

Vue.js konzentriert sich auf die Gestaltung von Benutzeroberflächen. Dabei wird ganz bewusst kein umfassendes Gesamtpaket angeboten. Stattdessen beschränkt sich der Kern der Vue-Bibliothek bewusst auf den View-Layer. Daher stammt auch der Name, der (im Englischen »vju:« ausgesprochen) ganz ähnlich klingt wie »View«. Durch diesen schlanken Ansatz kann Vue auch in bestehende Projekte leicht integriert werden.

Weitere Funktionalitäten können nach Wunsch durch die inkrementelle Hinzugabe weiterer Bibliotheken und Komponenten sowie die Nutzung moderner Tools ergänzt werden. Für den Anfang genügt jedoch die Referenzierung von Vue über einen CDN-Anbieter wie *jsDelivr*.

Bei der in den Codebeispielen referenzierten Version von Vue handelt es sich um einen Entwickler-Build. Er ist nicht so performant wie die Version für den produktiven Einsatz, enthält aber umfassendere Fehler- und Warnmeldungen für die Konsole. Für den Einsatz in produktiven Umgebungen sollte ein minifizierter Produktiv-Build von Vue (beispielsweise via *https://cdn.jsdelivr.net/npm/vue/dist/vue.min.js*) referenziert werden.

Sehen wir uns nun einfach ein erstes Beispiel an:

```
<html>
 <head>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js">
  </script>
 </head>
```

```
<body>
 <div id="app">
  {{ message }}
 </div>

 <script>
  var app = new Vue({
   el: '#app',
   data: {
    message: 'Hello Vue!'
   }
  })
 </script>
</body>
<html>
```

Dieser Code erzeugt im Browser die folgende Ausgabe:

```
Hello Vue!
```

Hiermit haben wir bereits eine erste Vue-Anwendung geschrieben. Was im ersten Moment lediglich wie das Rendern eines Strings aussieht, ist in Wirklichkeit viel mehr. Das DOM der HTML-Seite und die Daten von Vue sind nun miteinander verknüpft. Durch diese Abhängigkeit werden Änderungen an den Daten sofort im Browser sichtbar. Dieses deklarative Rendering ist eine der Kernfunktionen von Vue.

Speichern Sie den obigen Code also einfach in einer Datei mit der Endung *.html*, öffnen Sie diese Datei in Ihrem Browser und starten Sie anschließend die JavaScript-Konsole (in Chrome ein Rechtsklick auf eine beliebige Stelle und *Untersuchen* wählen). Wenn Sie nun den Befehl **`app.message = "Hello again!"`** in der Konsole eingeben, erscheint der neue Text direkt im Fenster des Browsers.

KAPITEL 2

Die Vue-Instanz

Die *Root-Instanz* ist das Herzstück jeder Vue-Anwendung. Im Beispiel aus dem vorherigen Kapitel stellt sie etwa das Datenattribut `message` bereit, dessen Wert mit der Template-Syntax im DOM gerendert wird. Eine typische Vue-Anwendung beginnt immer mit einer Vue-Instanz, die via `new Vue` erstellt wird.

```
var vm = new Vue({
 ...
})
```

Vue ist teilweise vom MVVM-Pattern (*Model-View-ViewModel*) inspiriert, weshalb für die Instanziierung oft der Variablenname `vm` (für Viewmodel) genutzt wird.

Bei der Instanziierung wird immer ein Objekt mit Optionen übergeben. Einen Großteil dieser Optionen werden wir in diesem Teil des Buchs besprechen. Eine komplette Auflistung findet sich in Teil III, »Vue-API«.

Von der Root-Instanz ausgehend kann sich eine Baumstruktur ergeben, die sich aus verschachtelten HTML-Elementen und weiteren Komponenten von Vue zusammensetzt. Diese wiederverwendbaren *Components* werden im zweiten Teil dieses Buchs besprochen. Vorab sei jedoch gesagt: Auch Components sind Vue-Instanzen und arbeiten bis auf einige wenige Ausnahmen mit dem gleichen Optionenobjekt wie die Root-Instanz der Vue-Anwendung.

Daten und Funktionen

Immer wenn eine neue Vue-Instanz erzeugt wird, werden zunächst alle Attribute aus dem Objekt `data` des übergebenen Optionenobjekts in das Reaktivsystem von Vue übertragen. Ändert sich später eines dieser Attribute, wird auch der View-Layer aktualisiert.

```
<div id="app">
 {{ firstName }} {{ lastName }} shot first!
</div>

var vm = new Vue({
 el: '#app',
 data: {
  firstName: "Han",
  lastName "Solo"
 }
})
```

Ändern sich in diesem Beispiel die Werte von `firstName` oder `lastName`, wird auch der View-Layer aktualisiert. Dabei ist es wichtig, zu verstehen, dass diese reaktive Funktionalität nur für die Attribute des Objekts `data` gilt, die bereits bei der Instanziierung existieren.

Die nachträgliche Zuweisung von **`vm.sidekick = "Chewbacca"`** über die JavaScript-Konsole des Browsers würde zwar das neue Attribut anlegen, Änderungen an `sidekick` hätten aber keine automatische Aktualisierung des Views zur Folge.

Falls die Funktionalität einer Vue-Anwendung also bestimmte Attribute voraussetzt, die aber erst später mit tatsächlichen Werten gefüllt werden, sollten diese Attribute mit entsprechenden Initialwerten direkt bei der Erstellung der Vue-Instanz definiert werden:

```
data: {
 message: '',
 maxTries: 0,
 showToolbar: false,
 users: [],
 error: null
}
```

Auf alle Attribute aus dem Objekt `data` kann über die Instanzvariable `vm` zugegriffen werden, etwa `vm.firstName`. Darüber hinaus bie-